Optimization and Kinematics (Pre Lecture)

Dr. Neil T. Dantam

CSCI-561, Colorado School of Mines

Spring 2019

Introduction

Optimization

- Optimization Problem:
  - Define objective $f(x)$ to minimize/maximize
  - Define acceptable-region for x
  - Find the "best" x
- Significant mathematical and software engineering in optimization techniques
- Often, convenient and efficient way to solve robotics problems
- Post DRC (2015): Becoming "standard" technique

Outcomes

- Know common forms of optimization problems:
  - Linear Program
  - Quadratic Program
  - Nonlinear least-squares
  - Sequential Quadratic program
- Pose robotic kinematics problems as optimization

Outline

Velocity Inverse Kinematics

Review of Least Squares Workspace Control

Given:
- ϕ_{act}: current configuration
- $\begin{bmatrix}\omega_{\text{ref}} & \dot{v}_{\text{ref}}\end{bmatrix}^T$: reference workspace velocity

Find: $\dot{\phi}_{\text{cmd}}$: joint velocity to achieve reference workspace velocity

Solution: Jacobian Pseudoinverse / Least squares:

- $\begin{bmatrix}\omega_{\text{ref}} \\ \dot{v}_{\text{ref}}\end{bmatrix} = \mathbf{J}\dot{\phi}_{\text{cmd}}$
- $\rightsquigarrow \dot{\phi}_{\text{cmd}} = \mathbf{J}^{+}\begin{bmatrix}\omega_{\text{ref}} \\ \dot{v}_{\text{ref}}\end{bmatrix}$

Minimizes $\left|\dot{\phi}_{\text{cmd}}\right|$ (joint velocity)

or $\left|\mathbf{J}\begin{bmatrix}\omega_{\text{ref}} \\ \dot{v}_{\text{ref}}\end{bmatrix} - \dot{\phi}_{\text{cmd}}\right|$ (workspace error)

Constrained Velocity Inverse Kinematics

Given:
- $\boldsymbol{\phi}_{\text{act}}$: current configuration
- $\dot{\boldsymbol{\phi}}_{\text{act}}$: current joint velocity
- $\begin{bmatrix}\omega_{\text{ref}} & \dot{v}_{\text{ref}}\end{bmatrix}^T$: reference workspace velocity
- $\boldsymbol{\phi}_{\max}$, $\boldsymbol{\phi}_{\min}$: Position limits
- $\dot{\boldsymbol{\phi}}_{\max}$, $\dot{\boldsymbol{\phi}}_{\min}$: Velocity limits
- $\ddot{\boldsymbol{\phi}}_{\max}$, $\ddot{\boldsymbol{\phi}}_{\min}$: Acceleration limits

Find: $\dot{\boldsymbol{\phi}}_{\text{cmd}}$: joint velocity to achieve reference workspace velocity,
such that:

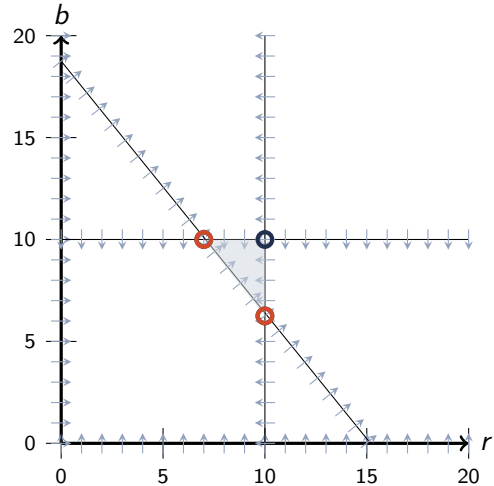
- $\boldsymbol{\phi}_{\max} \leq \boldsymbol{\phi} \leq \boldsymbol{\phi}_{\min}$
- $\dot{\boldsymbol{\phi}}_{\max} \leq \dot{\boldsymbol{\phi}} \leq \dot{\boldsymbol{\phi}}_{\min}$
- $\ddot{\boldsymbol{\phi}}_{\max} \leq \ddot{\boldsymbol{\phi}} \leq \ddot{\boldsymbol{\phi}}_{\min}$:

Example: Online Shopping

Buying beans b and rice r

Data beans: \$0.80/lb, rice: \$1.00/lb

- Buy more than 0 rice: $r \geq 0$
- Buy more than 0 beans: $b \geq 0$
- Over 10 rice would expire: $r \leq 10$
- Over 10 beans would expire: $b \leq 10$
- Need \$15 for free shipping, $.8b + r \geq 15$
- Minimize total cost: $.8b + r$



Linear Programming (LP)

Canonical Form

Definition: Linear Program

A linear program is an optimization problem that can be expressed in the following **canonical form**:

Maximize: $\mathbf{c}^T\mathbf{x}$

Subject to:
- $\mathbf{A}\mathbf{x} \leq \mathbf{b}$
- $\mathbf{x} \geq \mathbf{0}$

where:

- $\mathbf{x} \in \mathbb{R}^n$ is the variable to be optimized
- $\mathbf{c} \in \mathbb{R}^n$ is a constant vector for the linear objective function
- $\mathbf{A} \in \mathbb{R}^m \times \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^m$ are constants for the linear constraints

Linear Programming (LP)

"Extended" Form

Objective: **maximize** $\mathbf{c}^T\mathbf{x}$ or **minimize** $\mathbf{c}^T\mathbf{x}$

Subject to:

- $\mathbf{A}_{\max}\mathbf{x} \leq \mathbf{b}_{\max}$
- $\mathbf{A}_{\min}\mathbf{x} \geq \mathbf{b}_{\min}$
- $\mathbf{A}_{\text{eq}}\mathbf{x} = \mathbf{b}_{\text{eq}}$
- $\mathbf{x} \geq \mathbf{x}_{\min}$
- $\mathbf{x} \leq \mathbf{x}_{\max}$

Where:

- $\mathbf{x} \in \mathbb{R}^n$ is the variable to be optimized
- $\mathbf{c} \in \mathbb{R}^n$ is constant vector for a linear objective
- $\mathbf{A}_{\max}$, $\mathbf{b}_{\max}$ are max constraints
- $\mathbf{A}_{\min}$, $\mathbf{b}_{\min}$ are min constraints
- $\mathbf{A}_{\text{eq}}$, $\mathbf{b}_{\text{eq}}$ are equality constraints
- $\mathbf{x}_{\min}, \mathbf{x}_{\min}$ are constants for bounds on $\mathbf{x}$

Can covert to the canonical form.

Example: Standard Form

- $r \geq 0$
  $\rightsquigarrow -1 * r + 0 * b \leq 0$
- $b \geq 0$
  $\rightsquigarrow 0 * r + -1 * b \leq 0$
- $b \leq 10$
  $\rightsquigarrow 0 * r + 1 * b \leq 10$
- $r \leq 10$
  $\rightsquigarrow 1 * r + 0 * b \leq 10$
- $r + 0.8b \geq 15$
  $\rightsquigarrow -r + -.8 * b \leq -15$
- Minimize: $.8b + r$

Minimize: $\begin{bmatrix} 1 & .8 \end{bmatrix} \begin{bmatrix} r \\ b \end{bmatrix}$

Subject to:

$$\begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 0 & 1 \\ 1 & 0 \\ -1 & -.8 \end{bmatrix} \begin{bmatrix} r \\ b \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 10 \\ 10 \\ -15 \end{bmatrix}$$

Quadratic Programming (QP)

Canonical Form

Definition: Quadratic Program

A quadratic program is an optimization problem that can be expressed in the following **canonical form**:

$$\text{Minimize: } \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{c}^T\mathbf{x}$$

$$\text{Subject to: } \mathbf{A}\mathbf{x} \leq \mathbf{b}$$

where:

- $\mathbf{x} \in \mathbb{R}^n$ is the variable to be optimized
- $\mathbf{c} \in \mathbb{R}^n$ is a constant vector for the linear objective
- $\mathbf{Q} \in \mathbb{R}^n \times \mathbb{R}^N$ is a constant, symmetric matrix for the quadratic objective
- $\mathbf{A} \in \mathbb{R}^m \times \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^m$ are constants for the linear constraints

Quadratic Programming (QP)

"Extended" Form

Minimize: $\frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{c}^T\mathbf{x}$

Subject to:

- $\mathbf{A}_{\max}\mathbf{x} \leq \mathbf{b}_{\max}$
- $\mathbf{A}_{\min}\mathbf{x} \geq \mathbf{b}_{\min}$
- $\mathbf{A}_{\text{eq}}\mathbf{x} = \mathbf{b}_{\text{eq}}$
- $\mathbf{x} \geq \mathbf{x}_{\min}$
- $\mathbf{x} \leq \mathbf{x}_{\max}$

Where:

- $\mathbf{x} \in \mathbb{R}^n$ is the variable to be optimized
- $\mathbf{Q} \in \mathbb{R}^n$ is constant matrix for a quadratic objective
- $\mathbf{c} \in \mathbb{R}^n$ is constant vector for a linear objective
- $\mathbf{A}_{\max}$, $\mathbf{b}_{\max}$ are max constraints
- $\mathbf{A}_{\min}$, $\mathbf{b}_{\min}$ are mini constraints
- $\mathbf{A}_{\text{eq}}$, $\mathbf{b}_{\text{eq}}$ are equality constraints
- $\mathbf{x}_{\min}, \mathbf{x}_{\min}$ are constants for bounds on $\mathbf{x}$

Can covert to the canonical form.

Constrained Velocity Inverse Kinematics

Given:
- ϕ_{act}: current configuration
- $\dot{\phi}_{\text{act}}$: current joint velocity
- $\begin{bmatrix}\omega_{\text{ref}} & \dot{v}_{\text{ref}}\end{bmatrix}^T$: reference workspace velocity
- $\phi_{\max}$, $\phi_{\min}$: Position limits
- $\dot{\phi}_{\max}$, $\dot{\phi}_{\min}$: Velocity limits
- $\ddot{\phi}_{\max}$, $\ddot{\phi}_{\min}$: Acceleration limits

Find: ϕ_{cmd}: joint velocity to achieve reference workspace velocity, **such that no limits are violated**

Assume: We will apply $\dot{\phi}_{\text{cmd}}$ for time step Δt

Solution: Quadratic Program:

Minimize: $\left|\dot{\phi}_{\text{cmd}}\right|$

Subject to: $\mathbf{J}\dot{\phi}_{\text{cmd}} = \begin{bmatrix}\omega_{\text{ref}} \\ \dot{v}_{\text{ref}}\end{bmatrix}$ **and limit constraints**

Velocity IK as Quadratic Programming

Minimize: $\frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{c}^T\mathbf{x}$

Subject to: $\mathbf{A}_{\max}\mathbf{x} \leq \mathbf{b}_{\max}$ and $\mathbf{A}_{\min}\mathbf{x} \geq \mathbf{b}_{\min}$ and $\mathbf{A}_{\text{eq}}\mathbf{x} = \mathbf{b}_{\text{eq}}$

IK QP

Minimize:
$$\left(\dot{\phi}_{\text{cmd}}\right)^T \mathbf{Q} \left(\dot{\phi}_{\text{cmd}}\right)$$
$$\rightsquigarrow \left(\dot{\phi}_{\text{cmd}}\right)^T \mathbf{I} \left(\dot{\phi}_{\text{cmd}}\right)$$
$$\rightsquigarrow \left|\dot{\phi}_{\text{cmd}}\right|$$

Subject to: $\mathbf{J}\dot{\phi}_{\text{cmd}} = \begin{bmatrix}\omega_{\text{ref}} \\ \dot{v}_{\text{ref}}\end{bmatrix}$

Quantities

- $\phi_{\text{cmd}} \rightsquigarrow \mathbf{x}$:
  optimization variable
- $\mathbf{J} \rightsquigarrow \mathbf{A}_{\text{eq}}$:
  Equality constraint matrix
- $\begin{bmatrix}\omega_{\text{ref}} \\ \dot{v}_{\text{ref}}\end{bmatrix} \rightsquigarrow \mathbf{b}_{\text{eq}}$:
  Equality constraint vector

Velocity Constraints for IK QP

Velocity

- Limits: $\dot{\phi}_{\max}$, $\dot{\phi}_{\min}$
- Constraint:
  - $\dot{\phi}_{\text{cmd}} \leq \dot{\phi}_{\max}$ and
  - $\dot{\phi}_{\text{cmd}} \geq \dot{\phi}_{\min}$

$$\dot{\phi}_{\min} \leq \dot{\phi}_{\text{cmd}} \leq \dot{\phi}_{\max}$$

Position and Acceleration Constraints for IK QP

Position

- Limits: $\phi_{\max}$, $\phi_{\min}$
- Constraint:
  - $\phi_{\text{act}} + \Delta t \dot{\phi}_{\text{cmd}} \leq \phi_{\max}$ and
  - $\phi_{\text{act}} + \Delta t \dot{\phi}_{\text{cmd}} \geq \phi_{\min}$

$$\frac{\phi_{\min} - \phi_{\text{act}}}{\Delta t} \leq \dot{\phi}_{\text{cmd}} \leq \frac{\phi_{\max} - \phi_{\text{act}}}{\Delta t}$$

Acceleration

- Limits: $\ddot{\phi}_{\max}$, $\ddot{\phi}_{\min}$
- Constraint:
  $\ddot{\phi}_{\min} \leq \frac{\dot{\phi}_{\text{cmd}} - \dot{\phi}_{\text{act}}}{\Delta t} \leq \ddot{\phi}_{\max}$

$$\Delta t \ddot{\phi}_{\min} + \dot{\phi}_{\text{act}} \leq \dot{\phi}_{\text{cmd}} \leq \Delta t \ddot{\phi}_{\max} + \dot{\phi}_{\text{act}}$$

IK QP Summary

Minimize: $\left|\dot{\boldsymbol{\phi}}_{\text{cmd}}\right|$

Subject to:

- Reference constraint:

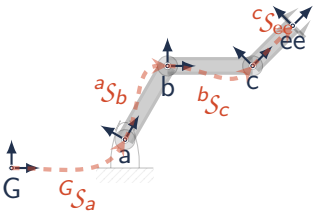
$$\mathbf{J}\dot{\boldsymbol{\phi}}_{\text{cmd}} = \begin{bmatrix} \omega_{\text{ref}} \\ \dot{v}_{\text{ref}} \end{bmatrix}$$

- Limit constraint: For each configuration i,

$$\max \begin{cases} \dot{\phi}_{\min,i} \\ \frac{\phi_{\min} - \phi_{\text{act}}}{\Delta t} \\ \Delta t \ddot{\phi}_{\min} \dot{\phi}_{\text{act}} \end{cases} \leq \dot{\phi}_{\text{cmd},i} \leq \min \begin{cases} \dot{\phi}_{\max,i} \\ \frac{\phi_{\max} - \phi_{\text{act}}}{\Delta t} \\ \Delta t \ddot{\phi}_{\max} + \dot{\phi}_{\text{act}} \end{cases}$$

Example: Arm QP

Overview (1/4)



Minimize: $\left| \begin{bmatrix} \dot{\phi}_a & \dot{\phi}_b & \dot{\phi}_c \end{bmatrix}^T \right|$

Subject to:

- Reference: $\mathbf{J} \begin{bmatrix} \dot{\phi}_a \\ \dot{\phi}_b \\ \dot{\phi}_c \end{bmatrix} = \begin{bmatrix} \omega_{\text{ref}} \\ \dot{v}_{\text{ref}} \end{bmatrix}$
- Limit:

$$\begin{bmatrix} b_{\min,a} \\ b_{\min,b} \\ b_{\min,c} \end{bmatrix} \leq \begin{bmatrix} \dot{\phi}_a \\ \dot{\phi}_b \\ \dot{\phi}_c \end{bmatrix} \leq \begin{bmatrix} b_{\max,a} \\ b_{\max,b} \\ b_{\max,c} \end{bmatrix}$$

Given: ϕ_{act}, $\dot{\phi}_{\text{act}}$, $\begin{bmatrix} \omega_{\text{ref}} & \dot{v}_{\text{ref}} \end{bmatrix}^T$, $\phi_{\max}$, $\phi_{\min}$, $\dot{\phi}_{\max}$, $\dot{\phi}_{\min}$, $\ddot{\phi}_{\max}$, $\ddot{\phi}_{\min}$

Find: $\phi_{\text{cmd}} = \begin{bmatrix} \dot{\phi}_a & \dot{\phi}_b & \dot{\phi}_c \end{bmatrix}^T$

Example: Arm QP

continued – Reference Constraint (2/4)

$$\mathbf{J} \begin{bmatrix} \dot{\phi}_a \\ \dot{\phi}_b \\ \dot{\phi}_c \end{bmatrix} = \begin{bmatrix} \omega_{\text{ref}} \\ \dot{v}_{\text{ref}} \end{bmatrix}$$

$$\overbrace{\begin{bmatrix} {}^{G}f_a \otimes \hat{\boldsymbol{k}} \otimes {}^{G}f_a^* & {}^{G}f_b \otimes \hat{\boldsymbol{k}} \otimes {}^{G}f_b^* & {}^{G}f_c \otimes \hat{\boldsymbol{k}} \otimes {}^{G}f_c^* \\ \mathbf{j}_{r_a} \times \left({}^{G}\mathbf{v}_{\text{ee}} - {}^{G}\mathbf{v}_a\right) & \mathbf{j}_{r_b} \times \left({}^{G}\mathbf{v}_{\text{ee}} - {}^{G}\mathbf{v}_b\right) & \mathbf{j}_{r_c} \times \left({}^{G}\mathbf{v}_{\text{ee}} - {}^{G}\mathbf{v}_c\right) \end{bmatrix}}^{\mathbf{J}} \begin{bmatrix} \dot{\phi}_a \\ \dot{\phi}_b \\ \dot{\phi}_c \end{bmatrix} = \begin{bmatrix} \omega_{\text{ref}} \\ \dot{v}_{\text{ref}} \end{bmatrix}$$

Example: Arm QP

continued – Limit Constraint (3/4)

$$\begin{bmatrix} b_{\min,a} \\ b_{\min,b} \\ b_{\min,c} \end{bmatrix} \leq \begin{bmatrix} \dot{\phi}_a \\ \dot{\phi}_b \\ \dot{\phi}_c \end{bmatrix} \leq \begin{bmatrix} b_{\max,a} \\ b_{\max,b} \\ b_{\max,c} \end{bmatrix}$$

where

- $b_{\min,c} = \max\left(\overbrace{\left(\frac{\phi_{\min} - \phi_{\text{act}}}{\Delta t} \right)}^{\text{position}}, \overbrace{\left(\dot{\phi}_{\min,i} \right)}^{\text{velocity}}, \overbrace{\left(\Delta t \ddot{\phi}_{\min} \dot{\phi}_{\text{act}} \right)}^{\text{acceleration}} \right),$
- $b_{\max,c} = \min\left(\underbrace{\left(\frac{\phi_{\max} - \phi_{\text{act}}}{\Delta t} \right)}_{\text{position}}, \underbrace{\left(\dot{\phi}_{\max,i} \right)}_{\text{velocity}}, \underbrace{\left(\Delta t \ddot{\phi}_{\max} + \dot{\phi}_{\text{act}} \right)}_{\text{aceleration}} \right)$

Example: Arm QP

continued – Summary (4/4)

Minimize: $\left| \begin{bmatrix} \dot{\phi}_a & \dot{\phi}_b & \dot{\phi}_c \end{bmatrix}^T \right|$

Subject to:

- Reference: $\mathbf{J} \begin{bmatrix} \dot{\phi}_a \\ \dot{\phi}_b \\ \dot{\phi}_c \end{bmatrix} = \begin{bmatrix} \omega_{\text{ref}} \\ \dot{v}_{\text{ref}} \end{bmatrix}$
- Limit: $\begin{bmatrix} b_{\min,a} \\ b_{\min,b} \\ b_{\min,c} \end{bmatrix} \leq \begin{bmatrix} \dot{\phi}_a \\ \dot{\phi}_b \\ \dot{\phi}_c \end{bmatrix} \leq \begin{bmatrix} b_{\max,a} \\ b_{\max,b} \\ b_{\max,c} \end{bmatrix}$,

  where
  - $b_{\min,c} = \max\left(\left(\dot{\phi}_{\min,i} \right), \left(\frac{\phi_{\min} - \phi_{\text{act}}}{\Delta t} \right), \left(\Delta t \ddot{\phi}_{\min} \dot{\phi}_{\text{act}} \right) \right)$,
  - $b_{\max,c} = \min\left(\left(\dot{\phi}_{\max,i} \right), \left(\frac{\phi_{\max} - \phi_{\text{act}}}{\Delta t} \right), \left(\Delta t \ddot{\phi}_{\max} + \dot{\phi}_{\text{act}} \right) \right)$

Outline

Position Inverse Kinematics

Given:
- ϕ_{act}: current joint configuration
- $\left({}^{G}\mathcal{S}_{e}\right)_{\text{ref}}$: reference workspace pose, i.e., orientation and translation as a dual quaternion

Find: ϕ_{cmd} such that ${}^{G}\mathcal{S}_{e}(\phi_{\text{cmd}}) = \left({}^{G}\mathcal{S}_{e}\right)_{\text{ref}}$, i.e., joint positions that achieve reference workspace pose

Nonlinear Least Squares (NLS)

Definition: Nonlinear Least-squares problem

A nonlinear least-squares problem is an optimization problem in the following form:

Minimize: $\left|\mathbf{r}\left(\mathbf{x}\right)^2\right| = \sum_{i=1}^{m} r_i\left(\mathbf{x}\right)^2$

where:
- $\mathbf{x} \in \mathbb{R}^n$ is the variable to be optimized
- $\mathbf{r} : \mathbb{R}^n \mapsto \mathbb{R}^m$ is the residual (error).

Gauss-Newton Algorithm

Overview

- **Given:** Initial guess $\mathbf{x}_{\text{start}}$
- **Repeat:**
  - Linearize $\mathbf{r}$ near the current point
  - Use linearized $\mathbf{r}$ to update point with the least-squares solution
- **Until** convergence

Gauss–Newton Algorithm

General Form

Procedure gauss-newton($\mathbf{x}_{\text{start}}, \mathbf{r}$)

Input: $\mathbf{x}_{\text{start}} \in \mathbb{R}^n$; // Initial guess for x

Input: $\mathbf{r} : \mathbb{R}^n \mapsto \mathbb{R}^m$; // residual function

1 $\mathbf{r}_{\text{start}} \leftarrow \mathbf{r}\left(\mathbf{x}_{\text{start}}\right)$; // Residual at $\mathbf{x}_{\text{start}}$

2 **if** $\left|\mathbf{r}_{\text{start}}\right| \leq \epsilon$ **then** // Small enough residual

3 **return** $\mathbf{x}_{\text{start}}$;

4 **else**

5 $\mathbf{J} \leftarrow \frac{\partial \mathbf{r}}{\partial \mathbf{x}}(\mathbf{x})$; // Jacobian

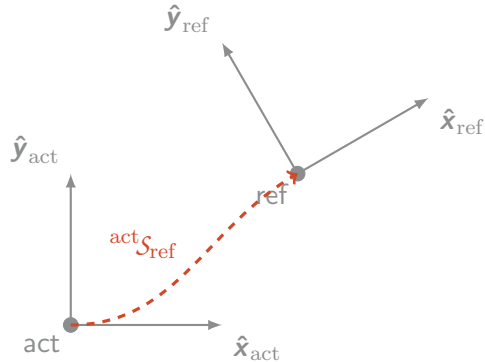
6 $\mathbf{x}_{\text{next}} \leftarrow \mathbf{x}_{\text{start}} + \overbrace{\left(\mathbf{J}^T \mathbf{J}\right)^{-1} \mathbf{J}^T}^{\text{Pseudoinverse: } \mathbf{J}^+} \mathbf{r}_{\text{start}}$;

7 **return** gauss-newton($\mathbf{x}_{\text{next}}, \mathbf{r}$)

Pose Error and Residuals

$$ {}^{G}\mathcal{S}_{\text{act}} \otimes {}^{\text{act}}\mathcal{S}_{\text{ref}} = {}^{G}\mathcal{S}_{\text{ref}} $$

1. $\begin{bmatrix} h_{\text{act}} \otimes h_{\text{err}} \\ \vec{v}_{\text{act}} + \vec{v}_{\text{err}} \end{bmatrix} = \begin{bmatrix} h_{\text{ref}} \\ \vec{v}_{\text{ref}} \end{bmatrix}$
2. $\begin{bmatrix} h_{\text{err}} \\ \vec{v}_{\text{err}} \end{bmatrix} = \begin{bmatrix} h^{*}_{\text{act}} \otimes h_{\text{ref}} \\ -\vec{v}_{\text{act}} + \vec{v}_{\text{ref}} \end{bmatrix}$
3. $\begin{bmatrix} \omega_{\text{err}} \\ \dot{\vec{v}}_{\text{err}} \end{bmatrix} = \begin{bmatrix} k_{\omega} \ln\left(h^{*}_{\text{act}} \otimes h_{\text{ref}}\right) \\ k_{v}\left(-\vec{v}_{\text{act}} + \vec{v}_{\text{ref}}\right) \end{bmatrix}$



IK Gauss–Newton Algorithm

Procedure ik-gauss-newton($\phi_{\text{start}}, \mathbf{r}, \mathbf{J}$)

Input: $\phi_{\text{start}} \in \mathbb{R}^n$; // Initial ϕ

Input: $\mathbf{r} : \mathbb{R}^n \mapsto \mathbb{R}^m$; // residual

Input: $\mathbf{J} : \mathbb{R}^n \mapsto \mathbb{R}^m \times \mathbb{R}^n$; // Jacobian

1 $\mathbf{r}_{\text{start}} \leftarrow \mathbf{r}(\phi_{\text{start}})$;

2 **if** $|\mathbf{r}_{\text{start}}| \leq \epsilon$ **then**

3 **return** ϕ_{start};

4 **else**

5 $\mathbf{J}_{\text{start}} \leftarrow \mathbf{J}(\phi_{\text{start}})$; // Evaluate

6 $\phi_{\text{next}} \leftarrow \phi_{\text{start}} + \left(\mathbf{J}^T\mathbf{J}\right)^{-1}\mathbf{J}^T\mathbf{r}_{\text{start}}$;

7 **return** ik-gauss-newton($\phi_{\text{next}}, \mathbf{r}, \mathbf{J}$)

Residual

$$\mathbf{r}(\phi) = \begin{bmatrix} \ln\left(\left(\hbar_{\text{act}}(\phi)\right)^* \otimes \hbar_{\text{ref}}\right) \\ -\vec{v}_{\text{act}}(\phi) + \vec{v}_{\text{ref}} \end{bmatrix}$$

Jacobian

$$\mathbf{J}(\phi) = \begin{bmatrix} \begin{pmatrix} \mathbf{j}_r \\ \mathbf{j}_p \end{pmatrix}_1 & \cdots & \begin{pmatrix} \mathbf{j}_r \\ \mathbf{j}_p \end{pmatrix}_n \end{bmatrix}$$

Prismatic: $\begin{pmatrix} \mathbf{j}_r \\ \mathbf{j}_p \end{pmatrix}_i = \begin{pmatrix} \mathbf{0} \\ {}^G\mathbf{u}_i \end{pmatrix}$

Revolute: $\begin{pmatrix} \mathbf{j}_r \\ \mathbf{j}_p \end{pmatrix}_i = \begin{pmatrix} {}^G\mathbf{u}_i \\ {}^G\mathbf{u}_i \times \left({}^G\mathbf{v}_e - {}^G\mathbf{v}_i\right) \end{pmatrix}$

Levenberg–Marquardt Algorithm

Procedure levenberg–marquardt($\mathbf{x}_{\text{start}}, \mathbf{r}, k$)

1 $\mathbf{r}_{\text{start}} \leftarrow \mathbf{r}\left(\mathbf{x}_{\text{start}}\right)$; // Residual at $\mathbf{x}_{\text{start}}$

2 **if** $\left|\mathbf{r}_{\text{start}}\right| \leq \epsilon$ **then return** $\mathbf{x}_{\text{start}}$;

3 **else**

4 $\quad \mathbf{J} \leftarrow \frac{\partial \mathbf{r}}{\partial \mathbf{x}}(\mathbf{x}_{\text{start}})$; // Jacobian

Damped Pseudoinverse: $\mathbf{J}^{\dagger}$

5 $\quad \mathbf{x}_{\text{next}} \leftarrow \mathbf{x}_{\text{start}} + \overbrace{\left(\mathbf{J}^T\mathbf{J} + k\mathbf{I}\right)^{-1}\mathbf{J}^T}\ \mathbf{r}_{\text{start}}$;

6 $\quad$ **if** $\left|\mathbf{r}(\mathbf{x}_{\text{next}})\right| > \left|\mathbf{r}_{\text{start}}\right|$ **then** // Error increased

7 $\quad\quad$ **return** levenberg-marquardt($\mathbf{x}_{\text{start}}, \mathbf{r}, \text{increase}(k)$);

8 $\quad$ **else** // Error decreased

9 $\quad\quad$ **return** levenberg-marquardt($\mathbf{x}_{\text{next}}, \mathbf{r}, \text{decrease}(k)$);

Outline

Constrained Position Inverse Kinematics

Given:
- $\boldsymbol{\phi}_{\text{act}}$: current joint configuration
- $\left({}^{G}\mathcal{S}_{e}\right)_{\text{ref}}$: reference workspace pose
- $\boldsymbol{\phi}_{\min}$, $\boldsymbol{\phi}_{\max}$: joint position limits

Find: $\boldsymbol{\phi}_{\text{cmd}}$ such that:
- ${}^{G}\mathcal{S}_{e}(\boldsymbol{\phi}_{\text{cmd}}) = \left({}^{G}\mathcal{S}_{e}\right)_{\text{ref}}$, i.e., joint positions that achieve reference pose
- $\boldsymbol{\phi}_{\min} \leq \boldsymbol{\phi}_{\text{cmd}} \leq \boldsymbol{\phi}_{\max}$

Naïve Solution:
- GNA / LMA may overshoot limits: $\boldsymbol{\phi}_{\min} \not\leq \boldsymbol{\phi}_{\text{start}} + \mathbf{J}^{\dagger}\mathbf{r}_{\text{start}} \not\leq \boldsymbol{\phi}_{\max}$
- Clamp joint values at limits: $\phi_{\text{next},i} \leftarrow \text{clamp}(\phi_{\text{next},i}, \phi_{\min,i}, \phi_{\max,i})$

Issues:
- Jacobian inverse "pushing through" joint limit
- Nonsmooth search space may cause failures

Nonlinear Programming

Definition: Nonlinear Program

A nonlinear program is an optimization problem of the following form:

Minimize: $f(\mathbf{x})$

Subject to:
- $\mathbf{h}(\mathbf{x}) = \mathbf{0}$
- $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$

where:

- $\mathbf{x} \in \mathbb{R}^n$ is the variable to be optimized
- $f : \mathbb{R}^n \mapsto \mathbb{R}$ is the objective function
- $\mathbf{h} : \mathbb{R}^n \mapsto \mathbb{R}^m$ is the equality constraint
- $\mathbf{g} : \mathbb{R}^n \mapsto \mathbb{R}^p$ is the inequality constraint

Generalizes LP, QP, NLS

Sequential Quadratic Programming

- **Intuition:** Extend Newton-like methods to constrained optimization
- **Overview:** For step k of the algorithm:
  - Begin with $\mathbf{x}^{[k]}$
  - Solve a QP to find $\mathbf{x}^{[k+1]}$
  - Repeat until convergence

Quadratic Subproblem

- Need to construct the QP to solve at each step
- Constraint Linearization:

  Minimize: $\frac{1}{2}\mathbf{d}^T \left(\mathbf{Q}^{[k]}\right)\mathbf{d} + \left(\mathbf{c}^{[k]}\right)^T \mathbf{d}$

  Subject to:
  - $\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\mathbf{d} = -\mathbf{h}\left(\mathbf{x}^{[k]}\right)$
  - $\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\mathbf{d} \leq -\mathbf{g}\left(\mathbf{x}^{[k]}\right)$

  where $\mathbf{d} = \mathbf{x} - \mathbf{x}^{[k]}$
- Objective Linearization (OK when constraints are Linear):
  - Gradient: $\mathbf{c}^{[k]} = \nabla f(\mathbf{x})$
  - Hessian: $\mathbf{Q}^{[k]} = \mathbf{H}(f(\mathbf{x}))$

Linearization

- Can be difficult to find or expensive to compute analytic gradient and Hessian of f.
- **Finite difference** approximation:
  - For scalar function $\ell(x) : \mathbb{R} \mapsto \mathbb{R}$
  - Derivative definition: $\frac{d\ell}{dx}(x) = \lim_{h \to 0} \frac{\ell(x+h) - \ell(x)}{h}$
  - Finite difference approximation: $\frac{d\ell}{dx}(x) \approx \frac{\ell(x+\epsilon) - \ell(x)}{\epsilon}$
- **Gradient:** finite difference over each x_i
- **Hessian:**
  - Jacobian of Gradient: $\mathbf{H}(f(x)) = \frac{\partial \nabla f}{\partial x}$
  - More common: Iterative, "quasi-Newton" methods: BFGS, PSB, DFP, SR1...

Gradient by Finite Difference

Procedure finite-difference-gradient(f,x)

/* Approximate the gradient $\nabla f(\mathbf{x})$ by finite difference */

Input: $f : \mathbb{R}^n \mapsto \mathbb{R}$;

Input: $\mathbf{x} \in \mathbb{R}^n$;

Output: $\mathbf{d} \in \mathbb{R}^n$; // Gradient, ∇f

1 **for** $i = 1 \ldots n$ **do**

2 $\quad \mathbf{p} \leftarrow (x_1, \ldots, x_{i-1}, x_i + \epsilon, x_{i+1}, \ldots, x_n)$;

3 $\quad \mathbf{m} \leftarrow (x_1, \ldots, x_{i-1}, x_i - \epsilon, x_{i+1}, \ldots, x_n)$;

4 $\quad d_i \leftarrow \frac{f(\mathbf{p}) - f(\mathbf{m})}{2\epsilon}$; // i^{th} element of ∇f

5 **return** $\mathbf{d}$;

Jacobian by Finite Difference

Procedure finite-difference-jacobian($\mathbf{f}$,x)

/* Approximate the Jacobian $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x})$ by finite difference */

Input: $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$;

Input: $\mathbf{x} \in \mathbb{R}^n$;

Output: $\mathbf{J} \in \mathbb{R}^m \times \mathbb{R}^n$;

1 **for** $i = 1 \ldots n$ **do**

2 $\quad \mathbf{p} \leftarrow (x_1, \ldots, x_{i-1}, x_i + \epsilon, x_{i+1}, \ldots, x_n)$;

3 $\quad \mathbf{m} \leftarrow (x_1, \ldots, x_{i-1}, x_i - \epsilon, x_{i+1}, \ldots, x_n)$;

4 $\quad \mathbf{j}_i \leftarrow \frac{\mathbf{f}(\mathbf{p}) - \mathbf{f}(\mathbf{m})}{2\epsilon}$; // i^{th} column of $\mathbf{J}$

5 $\mathbf{J} \leftarrow \begin{bmatrix} \mathbf{j}_1 & \ldots & \mathbf{j}_n \end{bmatrix}$;

6 **return** $\mathbf{J}$;

Hessian by Finite Difference

Procedure finite-difference-hessian($\nabla f, x$)

/* Approximate the Hessian $\mathbf{H}(f) = \frac{\partial \nabla f}{\partial \mathbf{x}}$ by finite difference */

Input: $\nabla f : \mathbb{R}^n \mapsto \mathbb{R}^m$; // Gradient of f

Input: $\mathbf{x} \in \mathbb{R}^n$;

Output: $\mathbf{H} \in \mathbb{R}^m \times \mathbb{R}^n$;

1 $\mathbf{H} \leftarrow$ finite-difference-jacobian($\nabla f, x$);

2 **return** $\mathbf{H}$;

or solver uses a quasi-Newton (iterative) method.

Position IK SQP

Given:
- ϕ_{act}: current configuration
- $\left({}^{G}\mathcal{S}_{e}\right)_{\text{ref}}$: reference pose
- $\phi_{\min}$, $\phi_{\max}$: position limits

Find: ϕ_{cmd} such that:
- ${}^{G}\mathcal{S}_{e}(\phi_{\text{cmd}}) = \left({}^{G}\mathcal{S}_{e}\right)_{\text{ref}}$, i.e., achieves reference pose
- $\phi_{\min} \leq \phi_{\text{cmd}} \leq \phi_{\max}$

Solution: Solve the following SQP:

Minimize: pos-ik-obj(ϕ)

Subject to: $\phi_{\min} \leq \phi \leq \phi_{\max}$

Procedure pos-ik-obj(ϕ)

Input: $\phi \in \mathbb{R}^{n}$;

/* Forward kinematics */

1 $\mathcal{S}_{\text{act}} \leftarrow {}^{G}\mathcal{S}_{e}(\phi)$;

/* Pose error */

2 $\mathcal{S}_{\text{err}} \leftarrow \left(\mathcal{S}_{\text{act}}\right)^{*} \otimes \mathcal{S}_{\text{ref}}$;

/* Scalar error */

3 **return** $\left|\ln\left(\mathcal{S}_{\text{err}}\right)\right|$;

References

Papers:

- P. Beeson and B. Ames. "TRAC-IK: An open-source library for improved solving of generic inverse kinematics." Humanoids. IEEE, 2015.
- Z. Kingston, N. Dantam, and L. Kavraki. "Kinematically constrained workspace control via linear optimization." Humanoids. IEEE, 2015.
- S. Kuindersma, et al. "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot." Autonomous Robots 40.3 (2016).

Libraries:

- Commercial: Gurobi, CPLEX, SNOPT
- Open Source: GLPK, LPSolve, GLOP, CLP, NLopt